A Grid Resource Management Architecture

Draft

Steve Chapin
Syracuse University

Mark Clement Quinn Snell
Brigham Young University

Introduction

This white paper is a product of the 2nd Grid Forum meeting, in Chicago, Illinois, 19-21 October, 1999 [1]. At that meeting, the beginnings of a resource management architecture were discussed in the Scheduling and Resource Management working group [2]. After that meeting, we held a brainstorming session and outlined the architecture presented here, and offered for public comment. This is intended as a straw man to serve as the basis for discussion and as the starting point of a definition of a Grid Forum-endorsed architecture; it is *not* put forth as a *fait accompli*.

Definitions and Components

We used a few abstract definitions, which are intended to be inclusive rather than exclusive (e.g. we do not explicitly state that the tasks comprising a job may have dependencies between them; because we have not excluded that possibility, it is included). We have listed some augmentations of the base definitions, which may clarify or expound our meaning:

- A resource is something that can be used for a period of time.
  - Resources might or might not be renewable.
  - Resources have owners.
  - Owners may charge others for using resources.
  - Resources might be shared, and might be used exclusively.
  - Resources might be explicitly named (a single host) or part of an aggregate (8 nodes of a 16-node homogeneous cluster).
  - Examples: disk space, network bandwidth, specialized device time, CPU time
- A task is a consumer of resources.
  - This includes the traditional model of a computational task.
  - Also includes non-computational tasks such as file staging, communication.
- A job comprises a mixture of jobs or tasks.
  - There might or might not be dependencies (e.g. dataflow) between tasks.
  - Jobs can have recursive structure, meaning that jobs are composed of subjobs and/or tasks, and subjobs can themselves be decomposed further; tasks are the leaves in such a tree. The simplest job consist of a single task.
  - Simplest case: job has one task in it.
- A schedule is a mapping of tasks to resources.
  - Not jobs to resources, as jobs don't consume resources, tasks do.

    - This can include the notion of time.
- A job control agent is responsible for shepherding a job through the system
    - Acts as persistent control point for a job.
    - Coordinates between different components.
- A scheduler produces one or more schedules for an input list of jobs
    - The unit of granularity for scheduling is a job.
    - The scheduling can be subject to a set of constraints.
- A domain control agent can commit resources for use; as the name implies, the set of resources controlled by an agent is a control domain.
    - Schedulers are not necessarily domain control agents; they might not be able to commit the use of resources
    - This is what some people mean when they say local resource manager.
    - We expect, but cannot require, domain control agents to support reservations.
    - A domain control agent can provide state information, either through a publication service or via direct querying.
    - Control domains may contain schedulers.
    - Examples of domain control agents: Maui Scheduler, Globus GRAM, Legion Host Object
- An information service acts as a database for describing items of interest to the resource management systems, such as jobs and resources.
    - Note that we haven't described any particular access method or implementation; could be LDAP, could be a commercial database, could be something else.
    - The information service could also describe available schedulers, deployment agents, etc.
- A deployment agent implements schedules.
    - Plays Riker to the scheduler's Picard ("Make it so, number one!").
    - Probably includes negotiating with domain control agents for resources, etc.
- A user submits a job to the resource management system for execution.
- An admission agent decides when (if ever) a job submitted by a user enters the system.
    - It is an open question whether or not all submissions to the system must go through admission agents.
- A monitor tracks the progress of a job.
    - May interact with domain control agents.
    - May interact with schedulers to remap the job.

We have striven to be as general as is feasible in our definitions. Many of these distinctions are logical distinctions. For example, we have divided the responsibilities of schedulers, deployment agents, and monitors, although it is entirely reasonable and expected that some scheduling systems may combine two or all three of these in a single program. Schedulers outside control domains cannot commit resources; these are what we've been calling metaschedulers or super schedulers. In our early discussions, we intentionally referred to control domains as "the box" because it connotes an important separation of "inside the box" vs. "outside the box." Actions outside the box are requests; actions inside the box may be commands. It may well be that the

system is fractal in nature, and that entire grid scheduling systems may exist inside the box. The important point is that we can treat the control domain as a black box from the outside.

We have intentionally not defined any relationship between users, jobs, and the major entities in the system (admission agents, schedulers, deployment agents, and monitors). Possibilities range from per-user or per-job agents to a single monolithic agent per system; each approach has strengths and weaknesses, and nothing in our definitions precludes or favors a particular use of the system. We expect to see local system defaults (e.g. a default scheduler or deployment agent) with users substituting their personal agents when they desire to do so.

The reader will notice that we do not mention the word *queue* anywhere in our descriptions; queuing systems imply a homogeneity of resources and a degree of control that simply will not be present in true grid systems. Queuing systems will most certainly exist within control domains.

Interaction of Components

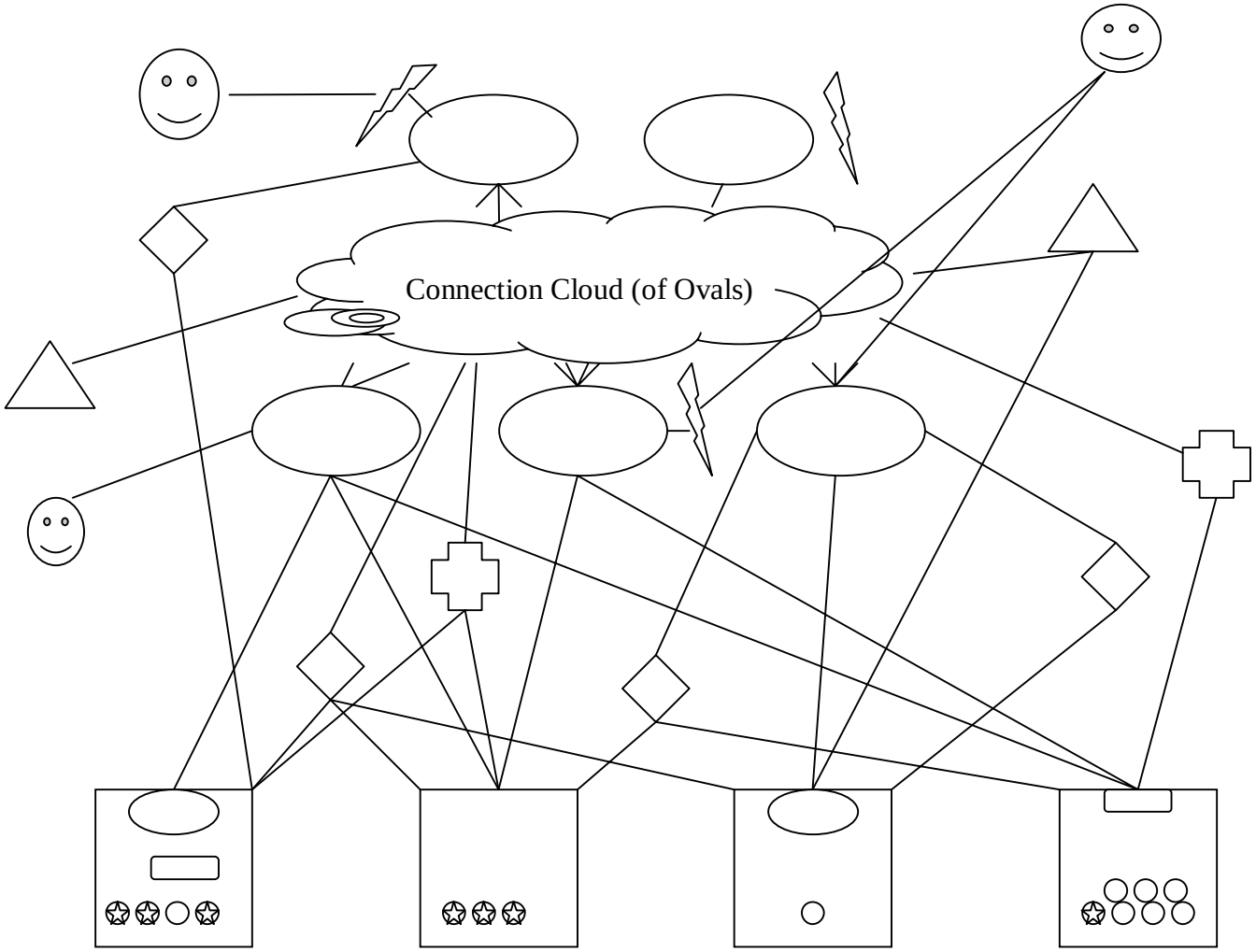
[NOTE: I am wondering if we need the notion of a job control agent, which is responsible for the overall control of a job. This agent might be the one the user hands the job to, and the JCA then submits it to an admission agent, then gets mappings from a scheduler (after getting permission from the admission agent), passes the job and mappings to a deployment agent, etc. This gives us a single control point for the job, which might be important. Should this be an optional or required portion of the architecture? Should we have a reservation agent as a separate entity from a deployment agent? I realized that I think of them together because that's how I built them in Legion, but it doesn't have to be that way.]

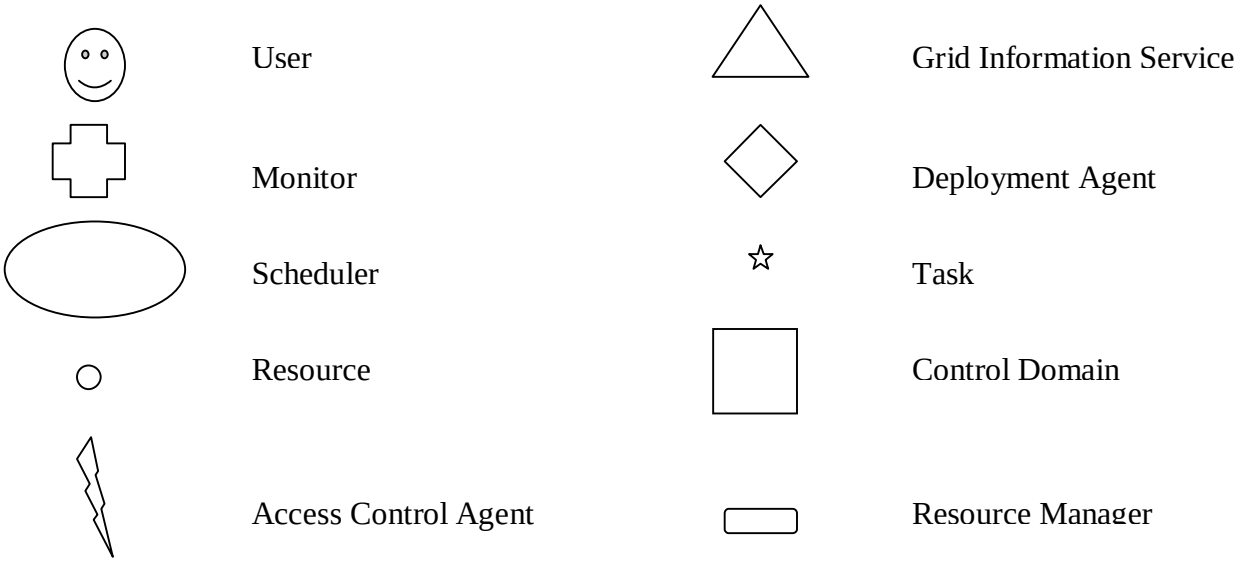
The figure below reflects the interactions between components of the resource management system. An arrow in the figure means that communication is taking place between components. We will next describe, at a high level, what we envision these interactions to be. This is the beginning of a protocol definition. Once the high-level operations are agreed upon, we can concern ourselves with wire-level protocols.

> We will begin with an example. Joe Random User submits a job to an admission agent. The admission agent examines the resource demands of the job (perhaps consulting with a grid information system) and determines that it is safe to add the job to the current pool of work for the system. The admission agent passes the job to a scheduler, which performs resource discovery using the grid information system and then consults with domain control agents to determine the current state and availability of resources.
>
> The scheduler then computes a set of mappings and passes these mappings to a deployment agent. The deployment agent negotiates with the domain control agents for the resources indicated in the schedule, and obtains reservations for the resources. These reservations are passed to the job control agent. At the proper time, the job control agent works with a different deployment agent, and the deployment agent coordinates with the appropriate domain control agents to start the tasks running. A monitor tracks progress of the job, and may later decide to reschedule if performance is lower than expected.

This is but one way in which these components might coordinate. Some systems will omit certain functionality (e.g. the job control agent), while others will combine multiple roles in a single agent. For example, a single process might naturally perform the roles of job control agent and monitor.

- Users: submit jobs to admission control agents (perhaps via job control agents)
- Schedulers: obtain information from grid information services and from DCA, get scheduling requests from job control agents or admission agents, pass mappings to deployment agents or JCA.
- Grid Information Service: service requests from schedulers, admission agents, etc.
- Admission agents: receive job requests from users or JCA, consult with GIS, report admission to JCA or by calling scheduler.
- Deployment agents: get reservations from DCA, get schedules from schedulers, report status of reservation/job start to JCA,
- Monitors: track status via DCA, report to JCA or scheduler.
- Job control agents: talk to just about everybody, as stated above.


Connection Cloud (of Ovals)


User
Grid Information Service
Monitor
Deployment Agent
Scheduler
Task
Resource
Control Domain
Access Control Agent
Resource Manager